Constrained Co-clustering of Gene Expression Data

Ruggero G. Pensa* Jean-François Boulicaut†

Abstract

In many applications, the expert interpretation of co-clustering is easier than for mono-dimensional clustering. Co-clustering aims at computing a bi-partition that is a collection of co-clusters: each co-cluster is a group of objects associated to a group of attributes and these associations can support interpretations. Many constrained clustering algorithms have been proposed to exploit the domain knowledge and to improve partition relevancy in the mono-dimensional case (e.g., using the so-called must-link and cannot-link constraints). Here, we consider constrained co-clustering not only for extended must-link and cannot-link constraints (i.e., both objects and attributes can be involved), but also for interval constraints that enforce properties of co-clusters when considering ordered domains. We propose an iterative co-clustering algorithm which exploits user-defined constraints while minimizing the sum-squared residues, i.e., an objective function introduced for gene expression data clustering by Cho et al. (2004). We illustrate the added value of our approach in two applications on gene expression data.

1 Introduction

In many application domains, the data analyst has to consider possibly large data sets that record numerical values of given properties for given objects (say $objects \times features$ matrices). From that perspective, the popular context of basket data analysis is the special case where boolean values record whether a product belongs to a transaction or not. In this paper, we will consider gene expression data sets that record gene expression values for given genes in given biological samples (see, e.g., microarray data analysis in [1]). A toy example data set is given in Fig. 1. In such a data set X_r, properties may denote biological samples and each object may be associated to one particular gene. For example, in X_r, one would say that the gene expression value for Gene 2 in Experiment 3 is 5.

Exploratory data analysis processes often make use of clustering techniques to get insights about global patterns within the data, i.e., to propose partitions of objects and/or of properties such that a grouping quality measure is optimized. Many clustering methods can compute partitions but suffer from the lack of explicit cluster characterization. This has motivated the research on conceptual clustering, e.g., the co-clustering approaches described in [24, 14, 2]. The objective of co-clustering is to compute co-clusters that are associations of (possibly overlapping) sets of objects with sets of properties. A co-clustering algorithm computes simultaneously linked partitions on both row and column dimensions. An example of a bi-partition in X_r would be $\{\{1,2,3,4,5\},\{6,7\}\}$ for objects, $\{\{1,4,5\},\{2,3\}\}$ for properties. This co-clustering indicates that the characterization of objects from $\{1,2,3,4,5\}$ is that they tend to share similar values from $\{1,4,5\}$. Also, properties in $\{2,3\}$ can be used to characterize objects in $\{6,7\}$. Co-clustering has been well studied in the context of gene expression data analysis because it provides valuable information about putative regulation mechanisms and biological functions [10, 21]. Intuitively, a co-cluster extracted from a gene expression matrix denotes a set of genes with similar expression profiles along its associated set of biological samples.

$$X_r = \begin{bmatrix} 3 & 0 & 0 & 2 & 4 \\ 1 & 4 & 5 & 1 & 2 \\ 4 & 1 & 0 & 4 & 5 \\ 2 & 0 & 1 & 3 & 4 \\ 1 & 4 & 5 & 1 & 2 \\ 1 & 4 & 6 & 0 & 0 \\ 0 & 5 & 6 & 0 & 0 \end{bmatrix}$$

Figure 1: A toy example X_r

We are interested in new co-clustering methods for enforcing the relevancy of computed bi-partitions in general and their application to gene expression data analysis in particular. Given a (co-)clustering algorithm, the analyst has generally a weak control on the clusters he/she obtains. Typically, he/she can decide for ad-hoc parameter settings which are quite operational and conceptually far from the declarative specification of desired properties. A co-clustering

*Pisa KDD Laboratory, ISTI-CNR, Pisa, Italy. ruggero.pensa@isti.cnr.it

†INSA-Lyon, LIRIS CNRS UMR5205, Villeurbanne, France. jean-francois.boulicaut@insa-lyon.fr

algorithm tries to optimize an objective function (e.g., Goodman-Kruskal's τ coefficient in [24] or the loss of mutual information in [14]) but it may also ensure that some user-defined constraints are satisfied (e.g., the fact that some objects and/or properties have to be together or not). Enforcing constraints can lead to lower values for the objective functions, and it is clear that combining objective function optimization and the satisfaction of other user-defined constraints is challenging.

The last 5 years, several researchers have studied mono-dimensional constrained clustering for simple types of user-defined constraints, mainly the so-called must-link and cannot-link constraints [25, 5, 18, 6, 13, 12, 7]. To the best of our knowledge, constrained co-clustering has been rarely studied. In [22, 23], we have studied co-cluster discovery when at least one of the dimensions is ordered and when interval constraints are defined w.r.t. orders. A typical application concerns kinetic gene expression data analysis. In this case, objects denote gene expression level measurements performed for successive time points. For an organism like Plasmodium Falciparum [9], we see that during its life cycle, groups of genes are activated and then inhibited, being somehow characteristic of development stages. Using interval constraint can support the discovery of such groups from experimental data.

In this paper, we introduce our constraint-based co-clustering approach which is quite different from the one in [22] extended in [23]. First, these papers concern only 0/1 data mining. More importantly, we propose here to work directly on the data, i.e., without any postprocessing of collections of local patterns that have to be computed beforehand (See Section 2). Our method builds a bi-partition that satisfies the user-defined constraints while optimizing the objective function introduced in [11], namely the sum-squared residues.

The rest of the paper is organized as follows. Section 2 discusses the related work. Section 3 is dedicated to the problem setting for constrained co-clustering. Section 4 provides details about our co-clustering algorithms. Section 5 is an empirical study about the added-value of our framework on two gene expression data set analysis. Section 6 briefly concludes.

2 Related work

Constrained co-clustering is a new approach to gene expression data analysis. To the best of our knowledge, only our previous work [22] has addressed the problem of co-clustering under user-defined constraints. We consider that optimizing the objective function and enforcing the number of co-clusters are more or less implicit constraints. In other terms, we would say that an algorithm like CoCluster [14] just performs co-clustering and not constrained co-clustering.

In [22, 23], we have proposed a local-to-global approach to build bi-partitions under interval constraints. This is performed by postprocessing collections of local patterns (e.g., closed sets) extracted from 0/1 data sets. The basic idea is to translate the interval constraint into a relaxation which can be enforced in the collection of local pattern. Then, it uses a k-means-based approach to obtain a partition of local patterns. Finally, such a partition can be post-processed to determine the co-clustering structure over the data. We also suggested possibilities to process co-clustering counterparts of the popular must-link and cannot-link constraints for standard clustering. The main difference of our contribution w.r.t. this previous work, is that here we compute co-clustering directly by alternatively computing clusters on columns and rows w.r.t. a common objective function. A second difference is that our current proposal ensures, when needed, the satisfaction of the interval constraint on the computed bi-partition. This was not the case in the method in [22, 23]. Third, our new approach works on numerical matrices: it is not limited to 0/1 data anymore. Last, but not least, we significantly improve must-link and cannot-link constraint processing over both sets of objects and attributes. Other related works can be identified in the areas of constrained clustering, and available co-clustering techniques.

Constrained clustering Constrained clustering is a relatively new field of research. It has been mainly studied as one approach to semi-supervised learning[1]. Semi-supervised clustering can support classification tasks when labeled data are limited and/or expensive to collect. A solution is to use the knowledge given by available labeled instances within a clustering algorithm. In [25], the authors have proposed a simple adaptation of k-means which enforces must-link and cannot-link constraints during the clustering process. [5] proposes a constraint-based clustering approach which uses labeled data during the initialization and clustering steps. An example of metric-based approach is presented in [18], while [8] integrates both constraint-based and metric-based approaches in a k-means-like algorithm. In [6], the authors propose a probabilistic model for semi-supervised clustering, which also combines the two approaches. Other related work focuses on constraint feasibility on a k-means-like approach [13], and on an agglomerative hierarchical clustering approach [12]. In such a related work, the goal is to improve accuracy in classification when only few instances are labeled, whereas our goal is not to support prediction:

[1] Another approach is called the metric-based approach for which a metric is trained considering labeled data and then standard clustering is applied.

we are working within an unsupervised framework. For us, constraints are used to specify user expectation (say subjective interestingness) and thus to improve the relevancy of extracted results. In the case of interval constraints, there are no pair-wise constraints (such as must-link and cannot-link), but the user has only to specify if he/she wants intervals or not, without knowing if a particular object x is in the same cluster than another object y. Moreover, notice that, for us, constraints can be applied both on objects and attributes.

Other applications of constrained clustering are the so-called sensor network and k-anonymity problems. In both applications, a possible solution is to find compact clusters containing a balanced number of objects. In [3, 4], the authors propose algorithms to discover balanced clusters. Recently, [15] has proposed an algorithm which finds an a priori unspecified number of compact clusters under combinations of minimum significance constraints and minimum variance constraints. Similarly to these two approaches, our algorithms enable to define constraints which are more related to the shape of the clusters rather than to pairs of objects.

Co-clustering Many co-clustering methods have been developed, possibly dedicated to gene expression data analysis. Kluger et al. [19] propose a spectral co-clustering method. First, they perform an adequate normalization of the data set to accentuate co-clusters if they exist. Then, they consider that the correlation between two columns is better estimated by the expression level mean of each column w.r.t. a partition of the rows. The bi-partition is computed by the algebraic eigenvalue decomposition of the normalized matrix. Their algorithm critically depends on the normalization procedure. Dhillon et al. [14] and Robardet et al. [24] have considered the two searched partitions as discrete random variables whose association must be maximized. Different measures can be used. Whereas Cocluster [14] uses the loss in mutual information, Bi-Clust [24] uses Goodman-Kruskal's τ coefficient to evaluate the link strength between the two variables. In both algorithms, a local optimization method is used to optimized the measure by alternatively changing a partition when the other one is fixed. The main difference between these two approaches is that the τ measure is independent of the number of co-clusters and thus Bi-Clust can automatically determine the number of co-clusters.

Lazzeroni et al. [20] propose to consider each matrix value as a sum of variables. Each variable represents a particular phenomenon in the data and corresponds to a co-cluster. In each co-cluster, column or row values are linearly correlated. Then, the method consists in determining the model minimizing the Euclidean distance between the matrix and the modeled values. This method is similar to the eigenvalue decomposition used in [19] without the orthogonal constraint on the computed variables.

In the context of gene expression data analysis, several authors have considered the computation of potentially overlapping local patterns that they call bi-clusters (see [21] for a survey). Ihmels et al. [17] propose a simple algorithm which builds in two steps a single association called a bi-cluster starting from a column set. First, they consider that the rows having a high score (greater than a threshold on the normalized matrix) on these columns belong to the bi-cluster. Then, they use the same principle to increase the original column set. In [10], Cheng et al. propose a co-clustering algorithm for numerical data. They define a bi-cluster as a subset of rows and subset of columns with a low mean squared residue. When the measure is equal to 0, the bi-cluster contains rows having the same value on the bi-cluster columns. When the measure is greater than 0, one can remove rows or columns to decrease the value. Thus the method consists in finding maximal size bi-clusters such that the measure is inferior to a threshold. Various heuristics can be used for this purpose. The same definition of residue is used in [11] to define the objective function which is also used in our current proposal. Authors propose two different residue measure, and show that the one proposed by Cheng et al. fits better to gene expression data analysis. Then, they introduce their co-clustering algorithm which optimizes the sum-squared residues function. This approach has been the starting point for our contribution.

Recently, Banerjee et al. [2], proposed a co-clustering formulation that is based on matrix approximation. The approximation error is measured using a large class of loss functions called Bregman divergences. The authors introduce a meta-algorithm whose special cases include the algorithms from [14] and [11].

3 A constrained co-clustering setting

Let $X \in \mathbb{R}^{m \times n}$ denote a data matrix. In the rest of the paper, the data set to be mined is the matrix X, and we always talk about rows and columns instead of objects and properties. Let x_{ij} be the element corresponding to row i and column j. For instance, x_{ij} might contain the expression level of gene i in the experimental condition j. Let $x_{i.}$ and $y_{.j}$ denote the vectors associated to, respectively, row i and column j.

A co-clustering $C^{k \times l}$ over X produces simultaneously a set of $k \times l$ co-clusters (a partition C^r into k groups of rows associated to a partition C^c into l groups of columns). To obtain a first quality criterion, we first

try to optimize a certain objective function.

DEFINITION 3.1. (OPTIMIZATION CONSTRAINT) *Let us assume an objective function $f(X, C^{k\times l})$, an optimization constraint $c_{opt}(f, X, C^{k\times l})$* is satisfied iff $C^{k\times l} = \text{argmin}_{\phi\in\mathcal{L}_{\mathcal{C}^{k\times l}}} f(X, \phi)$ *where $\mathcal{L}_{\mathcal{C}^{k\times l}}$ is the the collection of all possible co-clusterings.*

Some examples of objective functions are the Goodman-Kruskal's τ coefficient and the loss of mutual information [24, 14]. In this paper, we use the sum-squared residue function introduced in [11]. For computational feasibility reasons, co-clustering algorithms relax these optimization constraints, e.g., using local optimization heuristics. One can be interested in other kinds of constraints which are now defined.

DEFINITION 3.2. (MUST-LINK/CANNOT-LINK) *If rows i_a and i_b (resp. columns j_a and j_b) are involved in a **must-link** constraint, denoted $c_{=}(i_a, i_b)$ (resp. $c_{=}(j_a, j_b)$), they must be in the same cluster of $C^r = r_1, \ldots, r_k$ (resp $C^c = c_1, \ldots, c_k$). If rows i_a, i_b (resp. columns j_a and j_b) are involved in a **cannot-link** constraint, denoted $c_{\neq}(i_a, i_b)$ (resp. $c_{\neq}(j_a, j_b)$), they cannot be in the same cluster of $C^r = r_1, \ldots, r_k$ (resp $C^c = c_1, \ldots, c_k$).*

These forms of constraints have been studied, even in the semi-supervised clustering framework [8]. We generalize them to be able to apply them both to row sets and column sets. In an expression matrix, we can now exploit the knowledge about genes and/or experimental conditions. For example, if we know that gene i_a and gene i_b have the same function (say F) in the biological process, we can enforce a must-link constraint between these two genes to privilege the search for co-clusters associating genes having this function F and then identify a transcription module which could support such a biological function. We could also add some cannot-link constraints to avoid associations between experimental conditions which we would like to separate.

Let us now assume that a real value $s_c(j)$ (resp. $s_r(i)$) is associated to each column j (resp. row i). Then we have $s_r : \{1, 2, \ldots, m\} \to \mathbb{R}$ and $s_c : \{1, 2, \ldots, n\} \to \mathbb{R}$. For instance, $s_c(j)$ (resp. $s_r(i)$) could be a temporal or spatial measure related to j (resp. i). In microarray data, $s_c(j)$ might be the sampling time related to the DNA chip (say experiment) j. Another example would be to consider $s_r(i)$ as a measure of the absolute spacial position of a gene i in the DNA sequence of the studied organism. The two functions s_r and s_c enable to define an order $\preceq$ over the set of columns and/or rows. Indeed, we say that $j_a \preceq j_b$ iff $s_c(j_a) \leq s_c(j_b)$. In the rest of the paper, we say that, if a function s_c exists, then all the elements j are ordered, i.e., $\forall j_a, j_b$ s.t. $j_a < j_b$, $s_c(j_a) \leq s_c(j_b)$ (the same property holds for rows).

It could also be interesting to search for co-clusters which are coherent with the order defined by functions s_r and s_c. For instance, if we are interested in the different stages of the cell development, and we want to discover those genes that are mainly involved in each stage, we will look for clusters which are contiguous w.r.t. time. For this purpose, we can enforce an interval constraint.

DEFINITION 3.3. (INTERVAL CONSTRAINT) *If an order ($\preceq$) is defined over the column set (resp. row set), an **interval** constraint over this set, denoted $c_{int}(C^c)$, specifies that each cluster in C^c has to be an interval: $\forall c \in C^c$, if $j_a, j_b \in c$ then $\forall j_c$ such that $j_a \preceq j_c \preceq j_b$, $j_c \in c$.*

In general, the satisfaction of the must-link, cannot-link and interval constraints decreases the theoretical optimum of the objective function. We want a co-clustering algorithm which is able to take into account such constraints while trying to optimize the retained objective function. Notice that the satisfaction of a conjunction of constraints $c_{=}$, $c_{\neq}$ and c_{int} is not always feasible. For instance, for three objects i_1, i_2, i_3 such that $s(i_1) < s(i_2) < s(i_3)$, the conjunction $c_{=}(i_1, i_3) \wedge c_{\neq}(i_1, i_2) \wedge c_{int}(C^r)$ can never be satisfied, even though the sub-constraints of this conjunction do not cause any problem. In this paper, we assume that the processed conjunction of constraints is feasible. The constraint feasibility problem for both partitioning and hierarchical methods has been studied in [13, 12].

4 Using sum-squared residues

Our approach to constrained co-clustering is based on an iterative algorithm that minimizes the sum of squared residues. This objective function has been introduced in [11] for unconstrained co-clustering applied to gene expression data. It is an adaptation of a measure designed for local pattern discovery [10].

Given a data matrix $X \in \mathbb{R}^{m\times n}$, we search for a partition of X into k row clusters, and l column clusters. Let I be the set of indices of the rows belonging to a row cluster, and J the set of indices of the columns belonging to a column cluster. The sub-matrix of X determined by I and J is called a *co-cluster*. We use the definition of residue in [10].

DEFINITION 4.1. (RESIDUE) *Given an element x_{ij} of X, the residue of x_{ij} in the co-cluster defined by the sets of indices I and J, and whose respective cardinalities are*

$$X_r^1 = \left[\begin{array}{ccccc|c} 1 & 4 & 5 & 2 & 3 & \\ 3 & 2 & 4 & 0 & 0 & 1 \\ 1 & 1 & 2 & 4 & 5 & 2 \\ 4 & 4 & 5 & 1 & 0 & 3 \\ 2 & 3 & 4 & 0 & 1 & 4 \\ 1 & 1 & 2 & 4 & 5 & 5 \\ \hline 1 & 0 & 0 & 4 & 6 & 6 \\ 0 & 0 & 0 & 5 & 6 & 7 \end{array}\right] \quad X_r^2 = \left[\begin{array}{ccccc|c} 1 & 4 & 5 & 2 & 3 & \\ 3 & 2 & 4 & 0 & 0 & 1 \\ 4 & 4 & 5 & 1 & 0 & 3 \\ 2 & 3 & 4 & 0 & 1 & 4 \\ \hline 1 & 1 & 2 & 4 & 5 & 2 \\ 1 & 1 & 2 & 4 & 5 & 5 \\ 1 & 0 & 0 & 4 & 6 & 6 \\ 0 & 0 & 0 & 5 & 6 & 7 \end{array}\right] \quad X_r^3 = \left[\begin{array}{ccccc|c} 1 & 2 & 3 & 4 & 5 & \\ 3 & 0 & 0 & 2 & 4 & 1 \\ 4 & 1 & 0 & 4 & 5 & 3 \\ 2 & 0 & 1 & 3 & 4 & 4 \\ \hline 1 & 4 & 5 & 1 & 2 & 2 \\ 1 & 4 & 5 & 1 & 2 & 5 \\ 1 & 4 & 6 & 0 & 0 & 6 \\ 0 & 5 & 6 & 0 & 0 & 7 \end{array}\right]$$

Figure 2: Three co-clustering results on X_r (permutations on columns and rows to emphasize the co-clusters)

$|I|$ and $|J|$, is given by

$$h_{ij} = x_{ij} - x_{Ij} - x_{iJ} + x_{IJ} \tag{4.1}$$

where
$x_{IJ} = \frac{\sum_{i\in I, j\in J} x_{ij}}{|I|\cdot|J|}$
$x_{Ij} = \frac{\sum_{i\in I} x_{ij}}{|I|}$
$x_{iJ} = \frac{\sum_{j\in J} x_{ij}}{|J|}$.

Let $H = [h_{ij}] \in \mathbb{R}^{m\times n}$ denote the matrix of residues computed using the previous definition. The objective function to be minimized is the sum of squared residues [11] computed as follows:

$$||H||^2 = \sum_{I,J} ||h_{IJ}||^2 = \sum_{I,J} \sum_{i\in I, j\in J} h_{ij}^2 \tag{4.2}$$

We can rewrite the residue matrix in a more compact form. Let us introduce the matrices $R \in \mathbb{R}^{m\times k}$ and $C \in \mathbb{R}^{n\times l}$ which are defined as follows: each element (i,r) $(1 \le r \le k)$ of R is equal to $m_r^{-1/2}$ if i is in co-cluster r (m_r is the number of rows in r), 0 otherwise. Each element (j,c) $(1 \le c \le l)$ of the matrix C is equal to $n_c^{-1/2}$ if j is in c (n_c being the number of columns in c), 0 otherwise. The residue matrix becomes:

$$H = (I - RR^T)X(I - CC^T) \tag{4.3}$$

The proof of validity of this equation is given in [11]. The authors first demonstrate that $(RR^TX)_{ij} = x_{Ij}$, $(XCC^T)_{ij} = x_{iJ}$ and $(RR^TXCC^T)_{ij} = x_{IJ}$, before showing that (4.3) is true. They conclude that, if we consider the projection $(I - RR^T)X$ of the matrix X, then $||H||^2$ gives the objective function of k-means for this modified matrix.

Let us now consider our algorithmic contribution. Our approach uses the so-called "ping-pong" technique to process alternatively (applying a k-means method) columns and rows. Thus, matrix C is updated only after determining the nearest column cluster for each column (and similarly for rows). For that, we can decompose the objective function in terms of columns. Given $X^P = (I - RR^T)X$, $X^C = (I - RR^T)XC$, and $\hat{X}^P = (I - RR^T)XCC^T = X^CC^T$, we can rewrite the objective function as follows:

$$\begin{aligned} ||X^P - \hat{X}^P||^2 &= \sum_{c=1}^{l} \sum_{j\in J_c} ||X^P_{.j} - \hat{X}^P_{.j}||^2 \\ &= \sum_{c=1}^{l} \sum_{j\in J_c} ||X^P_{.j} - (X^CC^T)_{.j}||^2 \\ &= \sum_{c=1}^{l} \sum_{j\in J_c} ||X^P_{.j} - n_c^{1/2} X^C_{.c}||^2. \end{aligned}$$

In the same way, setting $X^P = X(I - CC^T)$, $X^R = R^TX(I - CC^T)$, and $\hat{X}^P = RR^TX(I - CC^T) = RX^R$, we obtain the following decomposition in terms of rows:

$$||X^P - \hat{X}^P||^2 = \sum_{r=1}^{k} \sum_{j\in I_r} ||X^P_{i.} - m_r^{1/2} X^R_{r.}||^2.$$

Then, matrices X^C and X^R correspond to the cluster centroids for columns and rows respectively.

Now we can introduce our constrained co-clustering algorithm. First, we give a version to solve the satisfaction problem for a conjunction of must-link and cannot-link constraints. Then, we introduce a version which processes the interval constraint. Finally, we sketch a possible strategy to integrate those two principles.

4.1 Satisfying must-link and cannot-link constraints The transitivity of the must-link constraint is a well known property. We can then transform a set of must-link constraints over rows into a collection $\mathcal{M}_r = M_1, \ldots, M_N$, where each M_i is a set of rows involved by the same transitive closure of must-link constraints. Let us denote $\mathcal{M}_c$ the same set built for columns and let $\mathcal{C}_r$ and $\mathcal{C}_c$ be the sets of cannot-link constraints for rows and columns respectively.

ALGORITHM 4.1. Co-clustering under must-link and cannot-link constraints.

ConsCoClust(X,k,l,$\mathcal{M}_r$,$\mathcal{M}_c$,$\mathcal{C}_r$,$\mathcal{C}_c$)
Input Data matrix X, k, l, cannot-link sets $\mathcal{C}_r$ and $\mathcal{C}_c$, collections $\mathcal{M}_r$ et $\mathcal{M}_c$
Output Matrices R and C
Initialize R et C
$\Delta = 1$; $\tau = 10^{-2}||X||^2$
$t = 0$
$obj^t = ||(I - RR^T)X(I - CC^T)||^2$
while $\Delta > \tau$
  $t = t + 1$
  $X^C = (I - RR^T)XC$
  $X^P = (I - RR^T)X$
  foreach $1 \leq j \leq n$
    $L = \emptyset$
    if $\exists M_v \in \mathcal{M}_c$ t.q. $j \in M_v$
      MLColumnAssign(X,l,L,M_v,$\mathcal{C}_c$)
    else
      $L = \{1 \leq c \leq l \mid \nexists j_c \mid \gamma^t[j_c] = c \wedge c_{\neq}(j, j_c) \in \mathcal{C}_c\}$
      $\gamma^t[j] = \text{argmin}_{c \in L} ||X^P_{.j} - n_c^{-1/2} X^C_{.c}||^2$
  Update C using γ
  $X^R = R^T X(I - CC^T)$
  $X^P = X(I - CC^T)$
  foreach $1 \leq i \leq m$
    if $\exists M_u \in \mathcal{M}_r$ t.q. $i \in M_u$
      MLRowAsssign(X,k,L,M_u,$\mathcal{C}_r$)
    $K = \emptyset$
    else
      $K = \{1 \leq r \leq k \mid \nexists i_r \mid \rho^t[i_r] = r \wedge c_{\neq}(i, i_r) \in \mathcal{C}_r\}$
      $\rho^t[i] = \text{argmin}_{r \in K} ||X^P_{i.} - m_r^{-1/2} X^R_{r.}||^2$
  Update R using ρ
  $obj^t = ||(I - RR^T)X(I - CC^T)||^2$
  $\Delta = |obj^t - obj^{t-1}|$

Algorithm 4.1 enables to co-cluster data when conjunctions of must-link et cannot-link constraints are given. It starts with some initialization (e.g., random initialization) of matrices C and R. During each iteration, the algorithm associates each column (resp. row) to the nearest column (resp. row) cluster which does not introduce any cannot-link violation. If a column (resp. row) is involved in a must-link constraint (see Algorithms 4.2 and 4.3), the algorithm associates the whole set of columns (resp. rows) involved in the transitive closure of this constraint to the column (resp. row) cluster such that the average distance is minimum, and controlling that there is no cannot-link constraint which is violated by this operation. Then the algorithm updates the matrix C (resp. R) following the assignment schema resulting from the previously described operations. This process is iterated until the diminution of the objective value turns to be smaller than a user-defined threshold τ.

ALGORITHM 4.2. Constrained column assignment.

MLColumnAssign(X,l,L,M_v,$\mathcal{C}_c$)
foreach $j_v \in M_v$
  $L = L \cup \{1 \leq c \leq l \mid \nexists j_c \mid \gamma^t[j_c] = c \wedge c_{\neq}(j_v, j_c) \in \mathcal{C}_c\}$
$\gamma^t[M_v] = \text{argmin}_{c \in L} \frac{\sum_{j_v \in M_v} ||X^P_{.j} - n_c^{-1/2} X^C_{.c}||^2}{|M_v|}$

ALGORITHM 4.3. Constrained row assignment.

MLRowAssign(X,k,L,M_u,$\mathcal{C}_r$)
foreach $i_u \in M_u$
  $K = K \cup \{1 \leq r \leq k \mid \nexists i_r \mid \rho^t[i_r] = r \wedge c_{\neq}(i_u, i_r) \in \mathcal{C}_r\}$
$\rho^t[M_u] = \text{argmin}_{r \in K} \frac{\sum_{i_u \in M_u} ||X^P_{i.} - m_r^{-1/2} X^R_{r.}||^2}{|M_u|}$

Notice that the initialization step should not necessarily take into account constraints, since their satisfaction is ensured by the first iteration of the algorithm. A possible improvement would consist in using a best assignment criterion for objects involved in cannot-link constraints. Moreover, we know that satisfying a set of cannot-link constraints for a given number of clusters is **NP**-complete [13].

4.2 Satisfying the interval constraint Algorithm 4.4 enables to solve the satisfaction problem for the interval constraint (the part concerning row assignment ($\star\star$) is omitted here). The initialization ($\star$) of partitions interested by this constraint should produce a number l (resp. k) of intervals over columns (resp. rows). Then, the assignment process only considers the frontiers between intervals. More precisely, it first processes the left frontier, then the right frontier iteratively. A column (resp. row) can be assigned to the adjacent interval if the distance is smaller than the distance computed over its original interval. In this case, we continue processing the remaining columns (resp. rows). When the left frontier and the right frontier of an interval correspond to the same column (resp. row), the algorithm starts to process the next frontier. If there is no necessity to reassign the column (resp. row), the algorithm stops the current frontier processing and it skips to the following one. Notice that, contrary to [22, 23], the satisfaction of the interval constraint on the computed bi-partition is here ensured.

We do not consider the combination of these two algorithms to process a conjunction of must-link, cannot-link and interval constraints. However, let us sketch research guidelines for this purpose. We first have to

ensure that each set $M_r \in \mathcal{M}_r$ (or $M_c \in \mathcal{M}_c$) is an interval. For instance, for a set of objects $\{i_1, i_2, i_3, i_4, i_5\}$, and a set $M_r = \{i_2, i_4\}$, we should include object i_3 into M_r because of the definition of intervals. Then, we need the initialization to produce a partition which takes into account the whole set of constraints (notice again that satisfying a conjunction of cannot-link constraints is a **NP**-complete problem). Finally, it is possible to reuse the strategy described by Algorithm 4.1 only on the frontiers, following the schema presented in Algorithm 4.4.

Notice that this approach is intrinsically different than applying existing mono-dimensional constraint-based clustering algorithms alternatively on row and column vectors. In fact, each column (row) reassignment step of the algorithm takes into account the previous row (column) reassignment step through the common objective function. As a consequence, constraints on one dimension potentially influence the partition on the other dimension.

4.3 A running example According to the objective function given by Equation 4.3, an optimal co-clustering result for X_r (see the toy example in Fig. 1), is given by matrix X_r^1 in Fig. 2. If we enforce Object 1 and Object 2 to be in two different clusters by setting a cannot-link constraint $c_{\neq}(1, 2)$, then we obtain the results shown by matrix X_r^2. Notice that, even if Object 5 is not concerned by this constraint, in the final bi-partition it is clustered together with Object 2. In fact, Object 2 and Object 5 share the same feature values.

If we set an interval constraint on the set of columns (namely $c_{int}(C^c)$), we obtain the co-cluster structure shown by X_r^3 in Fig. 2. Notice that, though only the column set is constrained, the resulting object partition is also different from the one obtained when no constraint is given. This is a direct consequence of using an objective function which takes into account both object and attribute partitions.

4.4 Computational complexity Let us first consider Algorithm 4.1. Computing $(I - RR^T)X(I - CC^T)$ only requires the number of operations needed to compute $R^T XC$, i.e., $kn(m + l)$. It gives $O(N)$ time complexity (when $N = mn$) under the reasonable hypothesis that $k \simeq l << m \simeq n$. Assigning columns and rows to the new clusters can be performed in $O(N(k + l))$ time at each iteration. The overall complexity of the algorithm is then $O(N(k+l)t)$, where t is the total number of iterations to complete the co-clustering. Complexity of Algorithm 4.4 is trivially the same. In general, the fact that we only process the interval frontiers increase the performances during the assignment step.

ALGORITHM 4.4. Co-clustering with an interval constraint.

IntCoClust(X,k,l)
Input Data matrix X, k and l, int_r, int_c
Output Matrices R and C
Initialize R, C, $left$, $right$; $(\star)$
$\Delta = 1$; $\tau = 10^{-2}||X||^2$
$t = 0$
$obj^t = ||(I - RR^T)X(I - CC^T)||^2$
while $\Delta > \tau$
  $t = t + 1$
  $X^C = (I - RR^T)XC$
  $X^P = (I - RR^T)X$
  foreach $1 \leq c \leq l$
    $finish = false$
    while $finish = false \wedge right[c] > left[c]$
      if $c > 1$
        $j = left[c]$
        if $||X^P_{.j} - n_{c-1}^{-1/2} X^C_{.c-1}||^2 <$
            $||X^P_{.j} - n_c^{-1/2} X^C_{.c}||^2$
          $\gamma^t[j] = c - 1$
          $left[c] = left[c] + 1$
          $right[c - 1] = right[c - 1] + 1$
        else
          $finish = true$
    $finish = false$
    while $finish = false \wedge right[c] > left[c]$
      if $c < l$
        $j = right[c]$
        if $||X^P_{.j} - n_{c+1}^{-1/2} X^C_{.c+1}||^2 <$
            $||X^P_{.j} - n_c^{-1/2} X^C_{.c}||^2$
          $\gamma^t[j] = c + 1$
          $left[c + 1] = left[c + 1] - 1$
          $right[c] = right[c] - 1$
        else
          $finish = true$
  Update C using γ
  $X^R = R^T X(I - CC^T)$
  $X^P = X(I - CC^T)$
  $(\star\star)$
  Update R using ρ
  $obj^t = ||(I - RR^T)X(I - CC^T)||^2$
  $\Delta = |obj^t - obj^{t-1}|$

5 Experimental validation

We have studied the impact of our constraint-based co-clustering approach on two well-documented microarray data sets, plasmodium [9] and drosophila [1]. The first one concerns the transcriptome of the intraerythrocytic developmental cycle of Plasmodium Falciparum, i.e., a causative agent of human malaria [9]. The data provide the expression profile of 3 719

genes in 46 biological samples. Each sample corresponds to a time point of the developmental cycle: it begins with merozoite invasion of the red blood cells, and it is divided into three main phases, the ring, trophozoite and schizont stages. These data, as well as functional groups descriptions, are available on ftp://ftp.camda.duke.edu/CAMDA04_DATASETS. Missing values (about 0.5%) were replaced by zeros.

The second data set concerns the gene expression of the Drosophila melanogaster during its life cycle [1]. The expression levels of 3 944 genes are evaluated for 57 sequential time periods divided into embryonic, larval and pupal stages. It has been retrieved from http://genome-www5.stanford.edu[2]. Missing values (less than 1%) were replaced by zeros.

Our algorithms are implemented in C, and all the experiments have been performed on a PC (Windows, Intel Core 2 Duo 2.00GHz, 2GB RAM). In all our experiments the value of the stopping parameter τ was set to $10^{-4}||X||^2$. As the initial partitions were randomly generated, our algorithm has been executed 20 times for each group of constraints. Running time is about 60 seconds for plasmodium data, and 100 seconds for drosophila.

5.1 Results for must-link et cannot-link constraints The first collection of experiments concerns the use of must-link and cannot-link constraints in two different situations. In the first one, we assume we know a certain amount of information about some functional groups of genes. We use it to set up a number of pairwise constraints, and then we measure the improvement in co-clustering results w.r.t. the unconstrained version of the same algorithm. In the second situation, we try to discover one fixed partition by using the co-cluster label to define some sets of pairwise constraints both on the gene set and the biological condition set.

Constraints on gene set We studied must-link and cannot-link constraints processing only for the gene set. For this purpose, we used the plasmodium data set, for which we can take advantage of a certain amount of information about genes involved in the different developmental stages. In particular, we considered the *cytoplasmic translation machinery* (denoted CTM) group (159 genes), which is known to be active during the first phase of the life cycle of the bacterium, the *merozoite invasion* (denoted MI) group (87 genes) particularly active during the second phase, and the *early ring transcripts* (denoted ERT) group (34 genes), which characterizes the last stage of the developmental cycle. All these functional groups have been described in [9]. We randomly selected 20 sets of constraints based on these well-described functional groups. Each set contains a variable number of constraints, and the amount of genes involved in each set is between 20% and 50% of the genes involved in the threes functional groups. For this experiment, we set $k = 3$ and $l = 3$ to be try to identify the three developmental stages of Plasmodium Falciparum.

	CTM	MI	ERT
Co-cluster1	**94.43%**	0.40%	20.44%
Co-cluster2	0.00%	**80.11%**	1.32%
Co-cluster3	5.57%	19.49%	**78.24%**

Table 1: Genes in the 3 clusters without constraints

	CTM	MI	ERT
Co-cluster1	**95.87%**	0.39%	19.26%
Co-cluster2	0.01%	**86.18%**	2.09%
Co-cluster3	4.12%	13.43%	**78.65%**

Table 2: Genes in the 3 clusters under constraints

The results presented in Table 1 and 2 show for each functional group the amount of genes involved in each co-cluster. In general, co-clusters are better characterized when constraints are defined. The number of genes belonging to each characterizing group increase in the co-cluster involving experimental conditions which are known to have this function activated. The improvement is much more significant for the second group of genes (merozoite invasion), while the last group (early ring transcripts) seems to be less influenced by the exploitation of these forms of constraints.

Constraint on gene and condition sets We measured the impact of combining constraints over row sets and column sets on the plasmodium data. For this purpose, we selected a bi-partition among the unconstrained co-clustering results. In particular, we chose the co-clustering results with the minimum objective function value obtained at the end of the iterative process. This value is about 1.99×10^4. Then, we generated 20 random sets of constraints involving gene and biological conditions. The number of genes involved in those constraints is about 5% to 10% of the total size of the gene set. For biological conditions, this number is about 15% to 25%. To evaluate the agreement between the selected bi-partition and the ones discovered by our constrained algorithm, we used the adjusted Rand index [16]. If $\mathbf{C} = \{C_1 \dots C_z\}$ is the partition built by

[2]The preprocessed and cleaned version of this data set is available on http://www-kdd.isti.cnr.it/~pensa/datasets/.

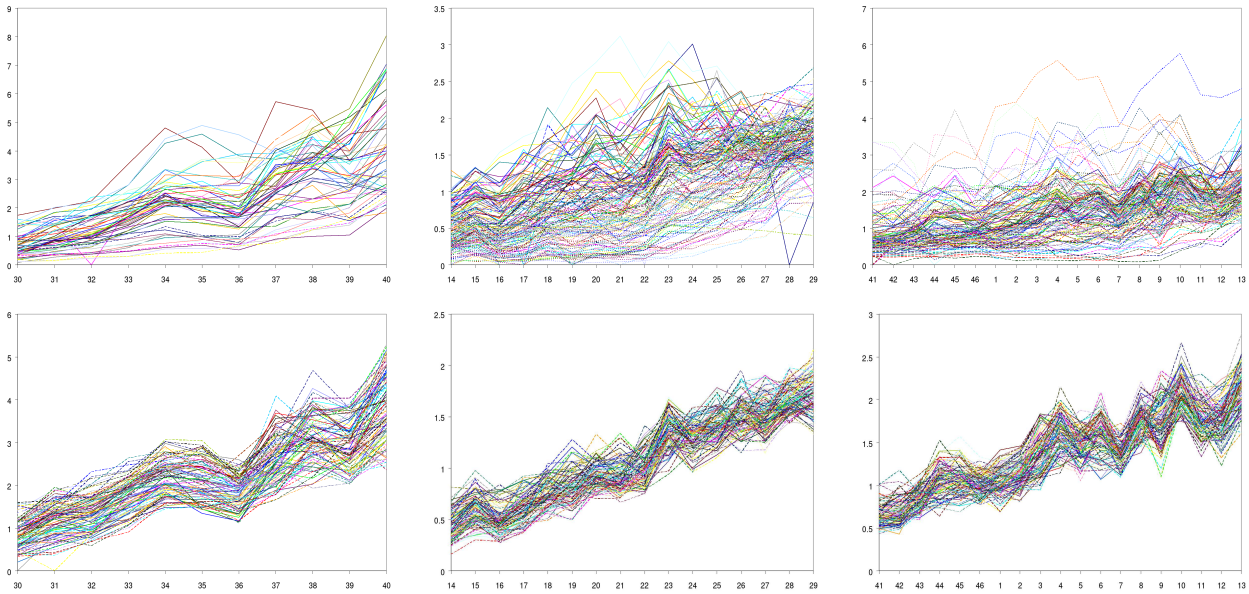

Figure 3: Two different views of 3 co-clusters captured with conjunctions of must-link and cannot-link constraints. The three top plots show the expression level of 100 randomly picked genes. Bottom plots show the expression level of the 100 genes closest to the average expression profile.

the clustering algorithm and $\mathbf{P} = \{P_1 \dots P_z\}$ is a predefined partition, each pair of points can be assigned to the same cluster or to two different clusters in each partition. Let a be the number of pairs belonging to the same cluster of $\mathbf{C}$ and to the same cluster of $\mathbf{P}$. The expected value of a denoted $exp(a)$ (p being the number of points) is computed as follows:

$$exp(a) = \frac{|\pi(C)| \cdot |\pi(P)|}{p(p-1)/2}$$

where

$$|\pi(C)| = \frac{\sum_{k=1}^{z} |C_k|(|C_k|-1)}{2}$$

$$|\pi(P)| = \frac{\sum_{k=1}^{z} |P_k|(|P_k|-1)}{2}.$$

Then, the maximum value for a is:

$$max(a) = \frac{1}{2}\left(|\pi(C)| + |\pi(P)|\right)$$

The agreement between $\mathbf{C}$ and $\mathbf{P}$ can be estimated as follows:

$$AR(\mathbf{C}, \mathbf{P}) = \frac{a - exp(a)}{max(a) - exp(a)}$$

Notice that when $AR(\mathbf{C}, \mathbf{P}) = 1$, we have identical partitions.

Results obtained by using constraints have been compared with those obtained by the unconstrained algorithm. Table 3 summarizes this experiment. We can see that using constraints gives rise to an obvious improvement of the agreement between the two bi-partitions (AR_r and AR_c are the agreement indexes measured on rows and columns respectively). As side effects, the average number of iterations needed by the algorithm to complete the co-clustering process is slightly smaller. Moreover, specifying constraints enables to improve the final value of the objective function. Indeed, the $||H||^2$ value decreases by about 2%.

	AR_r	AR_c	$\|\|H\|\|^2$	Nb.Iter
Const.	**0.88**	**0.73**	**2.16×10^4**	**9.18**
Unconst.	0.70	0.43	2.21×10^4	9.35

Table 3: Adjusted Rand index, final objective function value and number of iterations.

Examples of co-clusters discovered by our constrained techniques are given in Fig. 3. Even if gene clusters contain more than 1000 genes, it remains possible to identify similar expression profiles in the plots which show groups of 100 randomly picked genes for each co-clusters. If we consider the group of 100 genes which are the most similar with the average co-cluster, the related plots show more coherent expression profiles.

Notice that, since the sets of constraints are

randomly selected, these experiments include results achieved through constraints which can either positively influence the results or do not introduce any quality improvement. However, as average clustering results are good, we can conclude that our approach enables to obtain more relevant bi-partitions according to prior biological knowledge.

5.2 Results for the interval constraint We evaluated the added value of the interval constraint by applying our algorithm to the drosophila data set. Here, our goal is to rediscover the three phases of the drosophila life cycle using, as unique information, the number of clusters ($k = l = 3$).

	AR	$\|\|H\|\|^2$	Nb.Iter.
Const.	**0.76**	8.23×10^4	11.60
Unconst.	0.41	7.73×10^4	14.60
Const.Init.	0.54	7.86×10^4	**11.05**

Table 4: Adjusted Rand index, final objective function value and number of iterations.

We compared the adjusted Rand index for the constrained and unconstrained versions of our algorithm and for a collection of 20 randomly initialized runs. The results (see Table 4), show that using an interval constraint enables to find more accurately the three stages of the drosophila life cycle (the measured improvement for the adjusted Rand index is about 85%). Moreover, the number of iterations needed to complete the co-clustering process is considerably smaller than the one obtained when an interval constraint is enforced. Notice that the final value of the objective function for the unconstrained version of the algorithm is better than for the constrained version. It means that the structure which our algorithm is able to discover is unlikely to be the global optimum for this data set. Despite of this, the unconstrained algorithm has never managed to find intervals.

In Fig. 4, we plotted the average behavior of the objective function value w.r.t. iterations. For this type of data set, an interval-based initialization step gives rise to a smaller initial objective function. Interestingly, the objective function value for the unconstrained co-clustering algorithm starts to be better after the first iteration. Afterwards, the difference between the two curves is just an offset, while the two convergence speeds are quite similar. This example illustrates the tradeoff between the objective function optimization and the constraint satisfaction processes. We can comment such a process considering the metaphor of the "tug of war" game. When the only competitor is the objective

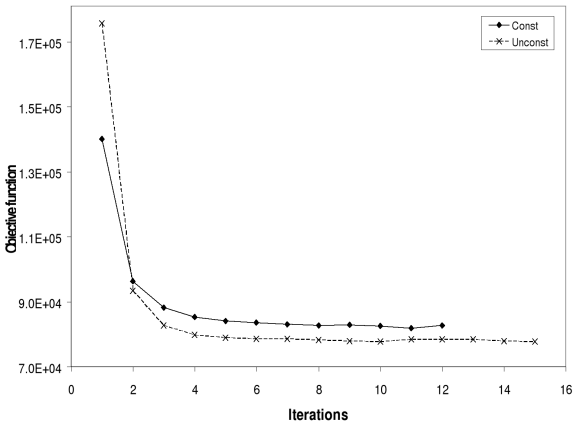

Figure 4: Object value vs. iterations

function, the only limit is its global optimum. When a second competitor (say constraints) plays, it reduces the objective function strength.

In order to measure the impact of the initialization method on the iterative behavior of the algorithm, we measured all the performances parameters already used for the previous comparison (see Table 4). Initializing the column partitions with a set of interval improve the average adjusted Rand index value over 20 runs, but this value is still far from the one achieved by the constrained version. The number of iteration is the only performance index which improves w.r.t. both unconstrained and constrained algorithms. Notice that, even if a constraint-oriented initialization has been used, none of the 20 executions has been able to find intervals.

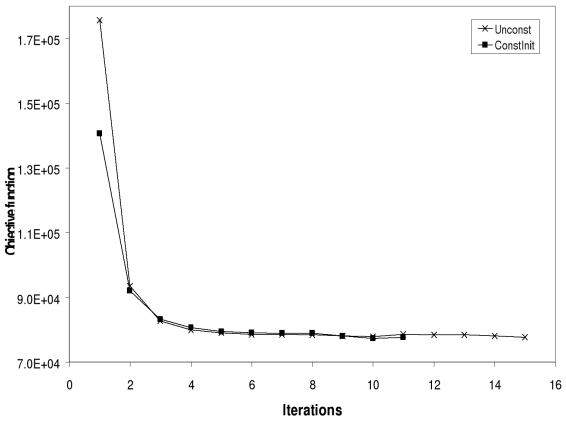

Figure 5: Object value vs. iterations

Finally, Fig. 5 illustrates that, after an improvement in the initial objective function, there are no significant differences between the totaly unconstrained algorithm

and the interval-initialized one. It means that the active constraint process introduced within Algorithm 4.4 is critical: a constraint-oriented initialization is not sufficient to satisfy the interval constraint.

6 Conclusion

Co-clustering is an interesting conceptual clustering approach. Improving co-cluster relevancy remains a difficult task in real-life exploratory data analysis processes. First, it is hard to capture subjective interestingness aspects, e.g., the analyst's expectation given her/his domain knowledge. Next, when these expectations can be declaratively specified, using them during the computational process of bi-partitions is challenging. In [22, 23], a simple approach was suggested that was dedicated to 0/1 data analysis and incomplete w.r.t. constraint processing. In this paper, we have proposed a new constrained co-clustering algorithm. We explained how to exploit user-defined constraints like must-link, cannot-link , and interval constraints when co-clustering numerical matrices. Applications on kinetic gene expression data analysis have been considered. Many other applications rely on ordered data analysis and may benefit from such constrained co-clustering approaches.

A short-term perspective is to combine properly the strategies for exploiting must-link and cannot-link constraints one one hand, and interval constraints on another hand. So far, our approach does not look for overlapping co-clusters while this may be interesting for many applications. We may study the possibility to discover overlapping co-clusters, like many local biclustering approaches already do (see, e.g., [10, 17]). Also, we are convinced that our approach can be easily extended towards other kinds of objective functions (e.g., mutual information [14] or the more general setting of Bregman functions [2]), and other user-defined constraints (e.g., balancing constraints [4]). Extending our algorithms and using them in other application domains (e.g., document clustering , privacy preserving clustering) is also planned.

Acknowledgements. The authors wish to thank the anonymous reviewers for their valuable suggestions. This research is partially funded by EU contracts IQ IST-FP6-516169 and GeoPKDD IST-FP6-014915.

References

[1] M.N. Arbeitman, E.E. Furlong, F. Imam, E. Johnson, B.H. Null, B.S. Baker, M.A. Krasnow, M.P. Scott, R.W. Davis, and K.P. White. Gene expression during the life cycle of drosophila melanogaster. *Science*, 297:2270–2275, september 2002.

[2] A. Banerjee, I. Dhillon, J. Ghosh, S. Merugu, and D.S. Modha. A generalized maximum entropy approach to bregman co-clustering and matrix approximation. *Journal of Machine Learning Research*, 8:1919–1986, 2007.

[3] A. Banerjee and J. Ghosh. On scaling up balanced clustering algorithms. In *Proceedings SIAM SDM 2002*, Arlington, VA, USA, 2002.

[4] A. Banerjee and J. Ghosh. Scalable clustering algorithms with balancing constraints. *Data Min. Knowl. Discov.*, 13(3):365–395, 2006.

[5] S. Basu, A. Banerjee, and R. J. Mooney. Semi-supervised clustering by seeding. In *Proceedings ICML 2002*, pages 27–34, Sydney, Australia, 2002.

[6] S. Basu, M. Bilenko, and R. J. Mooney. A probabilistic framework for semi-supervised clustering. In *Proceedings ACM SIGKDD 2004*, pages 59–68, Seattle, USA, 2004.

[7] S. Basu, I. Davidson, and K. Wagstaff (Editors). *Constrained Clustering: Advances in Algorithms, Theory and Applications*. Chapman & Hall/CRC Press, Data Mining and Knowledge Discovery Series, 2008. In press.

[8] M. Bilenko, S. Basu, and R. J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings ICML 2004*, pages 81–88, Banff, Canada, 2004.

[9] Z. Bozdech, M. Llinás, B. Lee Pulliam, E.D. Wong, J. Zhu, and J.L. DeRisi. The transcriptome of the intraerythrocytic developmental cycle of plasmodium falciparum. *PLoS Biology*, 1(1):1–16, October 2003.

[10] Y. Cheng and G. M. Church. Biclustering of expression data. In *Proceedings ISMB 2000*, pages 93–103, San Diego, USA, 2000. AAAI Press.

[11] H. Cho, I. S. Dhillon, Y. Guan, and S. Sra. Minimum sum-squared residue co-clustering of gene expression data. In *Proceedings SIAM SDM 2004*, Lake Buena Vista, USA, 2004.

[12] I. Davidson and S. S. Ravi. Agglomerative hierarchical clustering with constraints: Theoretical and empirical results. In *Proceedings PKDD 2005*, volume 3721 of *LNCS*, pages 59–70, Porto, Portugal, 2005. Springer.

[13] I. Davidson and S. S. Ravi. Clustering with constraints: Feasibility issues and the k-means algorithm. In *Proceedings SIAM SDM 2005*, Newport Beach, USA, 2005.

[14] I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. In *Proceedings ACM SIGKDD 2003*, pages 89–98, Washington, USA, 2003.

[15] R. Ge, M. Ester, W. Jin, and I. Davidson. Constraint-driven clustering. In *Proceedings of ACM SIGKDD 2007*, pages 320–329, San Jose, USA, 2007.

[16] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985.

[17] J. Ihmels, G. Friedlander, S. Bergmann, O. Sarig, Y. Ziv, and N. Barkai. Revealing modular organization in the yeast transcriptional network. *Nature Genetics*,

31:370–377, august 2002.

[18] D. Klein, S. D. Kamvar, and C. D. Manning. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In *Proceedings ICML 2002*, pages 307–314, Sydney, Australia, 2002.

[19] Y. Kluger, R. Basri, J. T. Chang, and M. Gerstein. Spectral biclustering of microarray data: coclustering genes and conditions. *Genome Research*, 13:703–716, 2003.

[20] L. Lazzeroni and A. Owen. Plaid models for gene expression data. technical report, stanford university, 2000.

[21] S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1(1):24–45, 2004.

[22] R. Pensa, C. Robardet, and J-F. Boulicaut. Towards constrained co-clustering in ordered 0/1 data sets. In *Proceedings ISMIS 2006*, volume 4203 of *LNCS*, pages 425–434, Bari, Italy, 2006. Springer.

[23] R. Pensa, C. Robardet, and J-F. Boulicaut. Constraint-driven co-clustering of 0/1 data. In *Constrained Clustering: Advances in Algorithms, Theory and Applications*. Chapman & Hall/CRC Press, Data Mining and Knowledge Discovery Series, 2008. In press.

[24] C. Robardet and F. Feschet. Efficient local search in conceptual clustering. In *Proceedings DS 2001*, volume 2226 of *LNCS*, pages 323–335, Washington, USA, november 2001. Springer.

[25] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl. Constrained k-means clustering with background knowledge. In *Proceedings ICML 2001*, pages 577–584, Williamstown, USA, 2001.